Job No.: 6065-89358
Converted from Japanese by the Ralph McElroy Translation Company
910 West Avenue, Austin, Texas 78701 USA

JAPANESE PATENT OFFICE
PATENT JOURNAL (A)
KOKAI PATENT APPLICATION NO. SHO 61[1986]-204741

Int. Cl.[4]:	G 06 F 9/44
Sequence No. for Office Use:	Z-8120-5B
Filing No.:	Sho 60[1985]-44477
Filing Date:	March 6, 1985
Publication Date:	September 10, 1986
No. of Inventions:	1 (Total of 4 pages)
Examination Request:	Not filed

SOURCE PROGRAM COMPRESSION METHOD

Inventor: Masayuki Higuma
NEC Corp.
5-33-1 Shiba
Minato-ku, Tokyo

Applicant: NEC Corp.
5-33-1 Shiba
Minato-ku, Tokyo

Agent: Yoshihiro Yahata, patent attorney

[There are no amendments to this patent.]

Claim

A source program compression method characterized by an encoding processing means and a storage means with areas for recording pre-stored reserved symbols in table form and variable symbols in table form, wherein the aforementioned encoding processing means categorizes the words constituting the source program into a group of reserved words for operators, a group of numerical value constants, a group of character string constants, and a group of variables, assigns corresponding word numbers to the words contained in the operator

reserved word group with reference to the aforementioned reserved symbol table, assigns corresponding word numbers to the words and encoded numerical values to the words contained in the numerical value constant group, assigns corresponding word numbers, numerical values indicating the lengths of the character strings, and the words for said character strings to the words contained in the character string constant group, records the words contained in the variable group in the variable symbol table, and adds the corresponding word numbers and the numbers indicating the locations they are registered in said symbol table to them for output.

Detailed explanation of the invention

Industrial application field

The present invention pertains to a source program compression method used for converting a source program into intermediate codes in an interpreter, such as a BASIC interpreter, widely utilized for a personal computer.

Prior art

Conventional interpreters, represented by the BASIC interpreter, e.g., include those in which the source program is interpreted directly with almost no conversion and in which it is interpreted after the data are converted into intermediate code for compression. However, a method in which only some of the operators, reserved words, and numerical value constants were encoded, and the other character string constants and variables were rewritten into intermediate code while they remained as character code used for external expression was used as the conventional intermediate encoding method.

Problems to be solved by the invention

In this conventional method, including not only the method in which there was almost no conversion of the source program but also the method utilizing intermediate encoding for compression, because while some of the operators, reserved words, and numerical value constants were encoded and the rest of the data remained as character codes for external expression, the compression rate relative to the data size of the source program was low.

In addition, although intermediate encoding was applied, because some portion remained as character codes for external expression, interpretation was to be carried out simultaneously with word analysis in order to execute a program of said type.

For example, assuming that up to 8 characters were allowed as the character length for a variable name, the word analysis required reading of character codes for up to 8 times, which imposed a restriction on the improvement of the execution speed. In addition, even where the processing was branched, as designated by a label such as the "goto*label," the label name was

rewritten by using the character codes for external expression as they were. Thus, it had to be interpreted by reading the character codes one at a time. Furthermore, when the destination of the jump indicated by the "goto*label" was located ahead of the line being executed, there was only the possibility of skipping the intermediate code character by character until the "goto*label" was encountered, so that a drop in the execution speed was inevitable.

In light of such conventional problems, the present invention presents a source program compression method with which the compression rate of intermediate codes can be improved in order specifically to improve the processing speed.

Means to solve the problems

The means for solving the aforementioned problems is a source program compression method characterized by an encoding processing means and a storage means with areas for recording a table of preregistered reserved symbols and a table of variable symbols, wherein the aforementioned encoding processing means categorizes the words constituting the source program into a group of reserved words for operators, a group of numerical value constants, a group of character string constants, and a group of variables, assigns corresponding word numbers to the words contained in the operator reserved word group with reference to the aforementioned reserved symbol table, assigns corresponding word numbers to the words and encoded numerical values to the words contained in the numerical value constant group, assigns corresponding word numbers, numerical values indicating the lengths of the character strings, and the words for said character strings to the words contained in the character string constant table, registers the words contained in the variable group in the variable symbol table, and adds the corresponding word numbers and the numbers indicating the locations they are recorded in said symbol table to them for output.

Application example

The present invention will be explained in detail on the basis of an application example. Figure 1 is a block diagram illustrating the concept of the present invention. Figure 2 is a flow chart showing an example in which the encoding processing means in Figure 1 is realized using a program. Figure 3 is an example of the source program. Figure 4 shows an example in which said source program is converted into intermediate code according to the aforementioned application example. In Figure 1, encoding processing means 2 obtains words one after another from source program 1, categorizes them, generates corresponding word numbers with reference to reserved symbol table 3 that is a table showing the correspondence between words and word numbers, and outputs them as intermediate code 5. That is, in Figure 2, encoding processing means 2 operates as follows. First, it obtains words one after another. Then,

1) when the word is a line number, it outputs its word number and the line number,

2) when the operator is a reserved word, it outputs each corresponding word number,

3) when the word is a numerical value constant, it outputs its word number and the numerical value converted into an internal expression,

4) when the word is a character string constant, it outputs its word number, character length, and character string, and

5) it records the word as a variable into the variable symbol table when it is not subsumed under aforementioned 1) through 4) and outputs its word number and the recorded position number.

According to the aforementioned operation, when the source program in Figure 3 is input in this example, because the number 100 at the beginning of the line is a line number, the corresponding word number (= 1) and line number (= 100) are output. Because the next SUM is not a line number, an operator, a reserved word, a numerical value constant, or a character string constant, it is recorded in the variable symbol table, and the word number (= 10) and registration location number (= 1) are output. Because next = is an operator, the corresponding word number (= 7) is output. The same operation is repeated thereafter in order to create and output the table shown in Figure 4 comprising intermediate codes and variable symbol table. A label is treated as a variable, recorded in the variable symbol table, and assigned with the word number and the registration location number corresponding to the variable.

Effects of the invention

As described above, the source program compression method of the present invention offers many advantages in that a high compression rate can be achieved because not only operators, reserved words, and numerical value constants but also variables and character string constants can be encoded.

Furthermore, all words, including variables (including labels), are assigned with word numbers, and the meanings of intermediate codes are interpreted immediately according to their word numbers. In particular, unlike in the prior art, the trouble of analyzing a word character by character in order to execute a label is no longer required, its contents can be understood immediately using the word number and the variable symbol table, so that the execution speed can be increased.

Brief description of the figures

Figure 1 is a block diagram showing the concept of the present invention. Figure 2 is a flow chart showing an example of the encoding processing means. Figure 3 is an example of the

source program. Figure 4 is a diagram for explaining an example in which the source program in Figure 3 is converted into intermediate codes.

1 ... source program; 2 ... encoding processing means; 3 ... reserved symbol table; 4 ... variable symbol table; and 5 ... intermediate code.

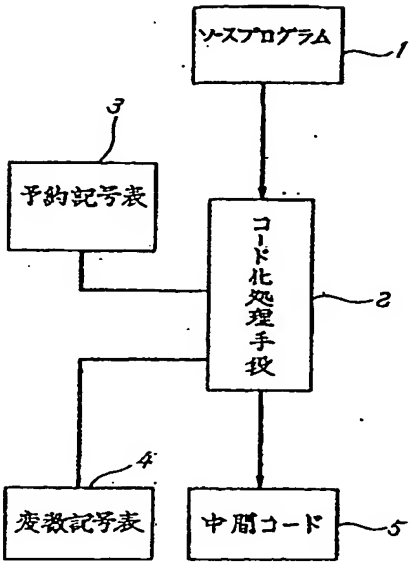


Figure 1. Block diagram illustrating the concept of the present invention

Key:
1 Source program
2 Encoding processing means
3 Reserved symbol table
4 Variable symbol table
5 Intermediate code

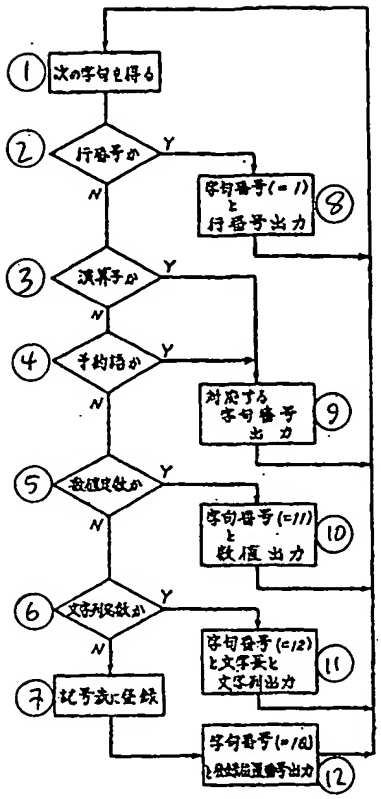


Figure 2. Flow chart showing an example of the encoding processing means

Key:
1 Obtain the next word
2 Is it a line number?
3 Is it an operator?
4 Is it a reserved word?
5 Is it a numerical value constant?
6 Is it a character string constant?
7 Register it into the symbol table
8 Output word number (= 1) and line number
9 Output corresponding word number
10 Output word number (= 11) and numerical value
11 Output word number (= 12), character length, and character string
12 Output word number (= 10) and registration location number

```
100  SUM=0:NUMBER=100
110  FOR  I=1 TO NUMBER
120     SUM=SUM+I
130  NEXT I
140  END
```

Figure 3. Example of the source program

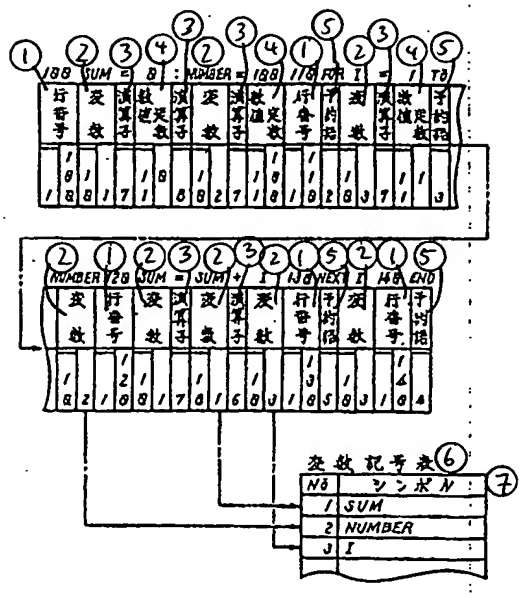


Figure 4. Diagram for explaining intermediate codes

Key:
1 Line number
2 Variable
3 Operator
4 Numerical value constant
5 Reserved word
6 Variable symbol table
7 Symbol

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　昭61－204741

⑤1Int.Cl.⁴　識別記号　庁内整理番号　⑬公開　昭和61年(1986)9月10日
G 06 F　9/44　Z－8120－5B

審査請求　未請求　発明の数 1　(全4頁)

⑤4発明の名称　ソースプログラムの圧縮方法

⑳特　願　昭60－44477
㉒出　願　昭60(1985)3月6日

⑦2発明者　日熊　政行　東京都港区芝5丁目33番1号　日本電気株式会社内
⑦1出願人　日本電気株式会社　東京都港区芝5丁目33番1号
⑦4代理人　弁理士　八幡　義博

明　細　書

1. 発明の名称

ソースプログラムの圧縮方法

2. 特許請求の範囲

コード化処理手段と、予め登録した予約記号表と変数記号表の登録領域を備えた記憶手段を有し、前記コード化処理手段はソースプログラムを構成する字句を演算子予約語群、数値定数群、文字列定数群と変数群に分類すると共に前記予約記号表をそれぞれ参照して演算子予約語群に属する字句には対応する字句番号を与え、数値定数群に属する字句には対応する字句番号とコード化した数値を与え、文字列定数群に属する字句には対応する字句番号と該文字列の長さを示す数値と該文字列の字句を与え、変数群に属する字句については該字句を変数記号表に登録しかつ対応する字句番号と該記号表上の登録位置番号を与えて出力することを特徴とするソースプログラムの圧縮方法。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明はパーソナルコンピュータにおいて広く利用されているBASICインタプリタなどのインタプリタにおいて、ソースプログラムを中間コードに変換する際のソースプログラムの圧縮方法に関する。

〔従来の技術〕

例えば従来のBASICインタープリタなどのインタープリタでは、ソースプログラムに殆んど変換を加えないで直接解釈実行するものや、中間コード化してデータを圧縮した後解釈実行するものなどがある。しかし、従来の中間コード化方法においては、一部の演算子、予約語および数値定数のみコード化し、それ以外の文字列定数および変数については、外部表現の文字コードのまま中間コードに転記する様な方法がとられていた。

〔発明が解決しようとする問題点〕

このような従来の方法においては、ソースプログラムに殆ど変換を加えないものは勿論、中

間コード化して圧縮する方法においても、一部の演算子、予約語および数値定数はコード化されていても、それ以外のものは外部表現の文字コードのままのデータであるので、ソースプログラムのデータの大きさと比べての圧縮率は小さい。

また、中間コード化されていても、このように一部が外部表現の文字コードのままであるため、このようなプログラムを実行する場合には、字句解析を行いつつ解釈実行する必要がある。

例えば、変数において、変数名の文字長が8文字まで許されているとすると、最大8回文字コードを読んで字句解析を行い実行することになるので、実行速度の向上のうえで制約となる。また、例えば「goto *ラベル」の如く、ラベルによつて分岐する処理の場合もラベル名を外部表現の文字コードのまま転記するような方法であつたため、これも一々文字コードを読んだ上で解釈実行することになり、更に「*ラベル」の飛び先が実行している行より後にある場合には中間コードを「*ラベル」が現れるまで1文字ずつ読み飛ばしていかざるを得ず、実行速度の低下は免れない。

本発明はこのような従来の欠点に鑑み、中間コードの圧縮率を上げ、特に処理速度の向上を可能にするソースプログラムの圧縮方法を提供するものである。

〔問題点を解決するための手段〕

前記問題点を解決するための手段は、コード化処理手段と、予め登録した予約記号表と変数記号表の登録領域を備えた記憶手段を有し、前記コード化処理手段はソースプログラムを構成する字句を演算子予約語群、数値定数群、文字列定数群と変数群に分類すると共に前記予約記号表をそれぞれ参照して演算子予約語群に属する字句には対応する字句番号を与え、数値定数群に属する字句には対応する字句番号とコード化した数値を与え、文字列定数群に属する字句には対応する字句番号と該文字列の長さを示す数値と該文字列の字句を与え、変数群に属する字句については該字句を変数記号表に登録しかつ対応する字句番号と該記号表上の登録位置番号を与えて、出力することを特徴とするソースプログラムの圧縮方法である。

〔実施例〕

本発明を実施例に基き詳細に説明する。第1図は本発明の概念を示すブロック図、第2図は第1図のコード化処理手段をプログラムで実現した場合の例を示すフローチャート、第3図はソースプログラムの例、第4図は該ソースプログラムを前記実施例に従つて中間コード化した例を示す。第1図において、コード化処理手段2はソースプログラム1から次々に字句を得てこれを分類し、字句と字句番号の対応表を有する予約記号表3を参照して対応する字句番号等を生成しこれを中間コード5として出力する。即ち、コード化処理手段2は第2図において、以下のように動作する。まず次々に字句を得て

1) 行番号であれば、字句番号と行番号を出力する。
2) 演算子が予約語であれば、各々に対応する字句番号を出力する。
3) 数値定数であれば字句番号と内部表現に変換した数値を出力する。
4) 文字列定数であれば、字句番号と文字長と該文字列を出力する。
5) 上記の1)～4)のいずれでもなければ変数として変数記号表に登録し、字句番号と登録位置番号を出力する。

以上のように動作するから、この例に第3図のソースプログラムを入力すると、行の先頭の数字100は行番号であるので、対応する字句番号(＝1)と行番号(＝100)が出力される。次のSUMは、行番号、演算子、予約語、数値定数、文字列定数のいずれでもないので、変数記号表に登録し、字句番号(＝10)と登録位置番号(＝1)が出力される。次の＝は演算子であるので、対応する字句番号(＝7)が出力され、以下同様の動作によつて、第4図に示すような中間コードと変数記号表が作成出力される。ラ

ベルについては、変数として取り扱われ、変数記号表に登録されると共に、変数に対応する字句番号と登録位置番号が与えられる。

〔発明の効果〕

以上のとおり本発明のソースプログラムの圧縮方法によれば、演算子、予約語、数値定数のみならず、変数、文字列定数についてもコード化するので圧縮率が大きい。

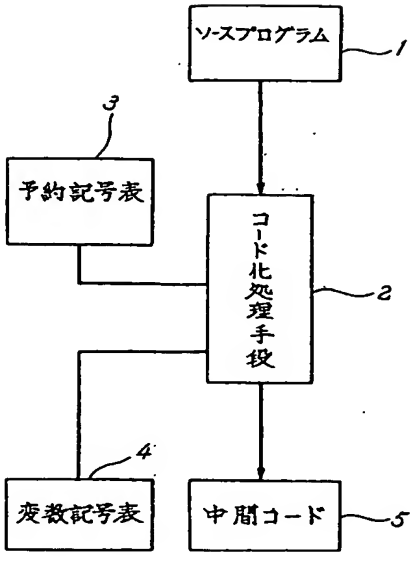
更に変数（ラベルを含む）等全ての字句について字句番号が与えられ、中間コードは字句番号によつて直ちにその意味が理解される。特に変数、ラベルについては従来の如く一文字ずつ読んで字句解析をしたうえ実行する手間がなく、字句番号と変数記号表によつて直ちにその内容を知ることができ、実行速度の高速化を図ることができるなどの多くの利点がある。

4. 図面の簡単な説明

第1図は本発明の概念を示すブロック図、第2図はコード化処理手段の例を示すフローチャート、第3図はソースプログラムの例、第4図は第3図のソースプログラムを中間コード化した例の説明図である。

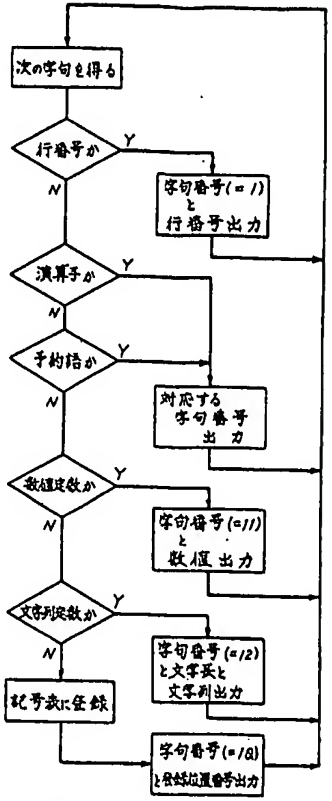
1 …… ソースプログラム　　2 …… コード化処理手段　　3 …… 予約記号表　　4 …… 変数記号表　　5 …… 中間コード

代理人 弁理士 八 幡 義 博



本発明の概念を示すブロック図

第 1 図



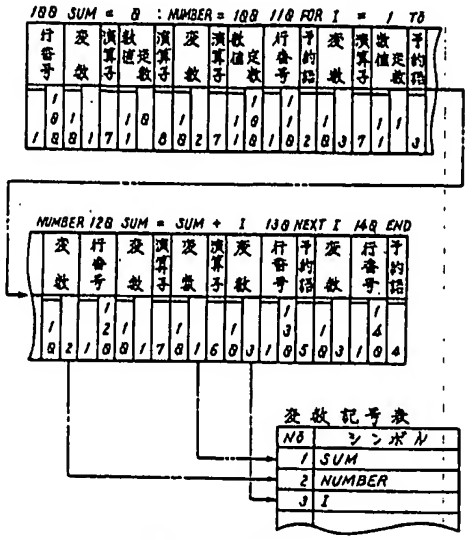
コード化処理手段の例のフローチャート

第 2 図

```
100  SUM = 0 : NUMBER = 100
110  FOR  I = 1  TO  NUMBER
120      SUM = SUM + I
130  NEXT  I
140  END
```

ソースプログラムの例

第 3 図



中間コードの説明図

第 4 図